Calculs de processus et verification

Gérard Boudol

INRIA

CENTRE
SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (3) 954 90 20

Rapports de Recherche

N° 424

CALCULS DE PROCESSUS ET VÉRIFICATION

Gérard BOUDOL

Juillet 1985

Calculs de Processus et Vérification

Gérard Boudol

INRIA Sophia-Antipolis
06565-Valbonne

Résumé

On présente brièvement la notion de calcul de processus parallèles, dont les termes sont interprétés comme des systèmes de transitions. La notion de congruence sur ces derniers induit des propriétés algébriques sur les termes, dont on montre l'utilisation à travers un exemple de vérification.

Abstract

We briefly introduce the notion of algebraic calculus of processes. The terms of calculi are interpreted as transition systems. The concept of congruence on these systems induces algebraic properties on terms, which are used in an example of verification.

1. Introduction

Il est reconnu depuis longtemps que la notion de fonction ne permet pas de donner une sémantique satisfaisante pour les systèmes parallèles et communicants. Ces systèmes manipulent certes des données et produisent des résultats, mais c'est *au cours des calculs* qu'un agent peut communiquer des données à un autre agent qui travaille parallèlement. La composition des fonctions, qui permet d'interpréter la composition séquentielle des programmes, est donc inapte à rendre compte de ce type de communication. Un modèle plus fin que la notion mathématique de fonction est par conséquent nécessaire.

Le modèle que nous adoptons ici repose sur l'idée de calcul d'une fonction: un tel calcul est une séquence de pas élémentaires, le déroulement d'un algorithme si l'on veut. Cette idée est d'ailleurs à la base de la sémantique opérationnelle des programmes séquentiels: un programme exécute une suite de changements d'états. On pourra trouver une description de ce type de sémantique dans le cours de G. Plotkin [14].

Les objets qui pour nous décrivent la sémantique des programmes parallèles et communicants sont donc des *systèmes de transitions:* un tel système, étant dans un certain état q, exécute un pas élémentaire de calcul a et, ce faisant, se reconfigure en un autre état q'. Cette relation de transition est notée:

$$q \xrightarrow{a} q'$$

Bien entendu, il faut s'entendre sur ce qu'est un "pas élémentaire de calcul"; pour nous cela veut simplement dire que pour un système (discret) considéré, il y a une notion d'action *atomique*, ou si l'on veut non-interruptible. En effet nous ferons ici dans une large mesure abstraction de la signification que peuvent avoir les actions effectuées au cours de transitions.

Il est usuel en mathématiques lorsqu'on s'intéresse à une classe de structures de définir

des notions de congruence et de morphisme. S'agissant de systèmes de transitions, le but essentiel est de pouvoir construire des "modèles réduits" (quotients) et comparer des systèmes. Dans la notion de congruence intervient une relativisation de l'atomicité: on se donne un point de vue abstrait sur les actions que peut exécuter un système, une action abstraite étant représentée par un ensemble de séquences d'actions "concrètes". Ainsi en construisant un quotient d'un système selon un tel critère d'abstraction, on en donne un modèle réduit qui permet de le visualiser en ne retenant que certains traits de son comportement. Un cas typique est celui où l'on décide de ne pas observer les communications internes, et de ne voir que le "service" rendu par un système. La notion de congruence est donc liée à la vérification de propriétés. De manière analogue, on peut par la notion de morphisme rendre compte de celle d' "implémentation": il s'agit là de traduire un système dans un autre en réalisant des actions abstraites par des successions d'actions de "plus bas niveau". Ces idées on été développées dans [4], et nous n'introduirons pas ici la notion de morphisme.

Poursuivant l'analogie avec ce qui a été fait pour la sémantique des programmes séquentiels, où l'on s'attache à définir la fonction calculée par un algorithme écrit avec certaines primitives, on peut se demander quelle *syntaxe* on peut se donner pour décrire les systèmes de transitions. Le type de syntaxe que nous adoptons ici est celui des algèbres de termes, où chaque opérateur représentera une façon de composer des systèmes. Cette "façon de composer" est *spécifiée* (complètement) par des règles qui indiquent quels peuvent être les comportements d'un système composite en fonction de l'activité de ses composants. Ce style de définition de la sémantique (dite "opérationnelle") est du à G. Plotkin [14].

On peut alors transposer au plan syntaxique les concepts introduits dans l'univers sémantique: une congruence dans la classe des systèmes constituée par l'interprétation d'une algèbre de termes induit une égalité sémantique entre ces termes. On peut donc raisonner directement dans la syntaxe, obtenant là un *calcul de processus* qui est:
-- une algèbre de termes
-- munie d'une sémantique opérationnelle, qui permet d'interpréter les termes comme des systèmes de transitions
-- munie aussi d'une égalité sémantique correspondant à une congruence sur les systèmes de transitions.
Cette notion de calcul de processus provient directement des travaux de R. Milner [10,12] (même si nous omettons des citations, il devrait être évident que nous ne faisons que prolonger la voie ouverte par Milner). La notion de morphisme quant à elle permet de parler de sous-calculs ou d'implémentation d'un calcul dans un autre (cf. [4]).

Ce que l'on gagne à considérer une syntaxe pour les systèmes de transitions, c'est que chaque congruence induit des propriétés sur les termes. Nous pouvons ainsi, par des raisonnements algébriques, conduire des *vérifications* de systèmes; pour nous, vérifier signifie donc prouver l'équivallence de termes, modulo un critère d'abstraction dans lequel s'exprime le "point de vue" adopté. Nous terminerons ces quelques notes en donnant de façon assez complète un exemple d'une telle preuve.

2. Systèmes de transitions

Comme on l'a dit précédemment, un système de transitions comporte des *états*, des *actions* et des *transitions*. On y ajoutera ici un *état initial*. On peut donc le décrire comme une structure

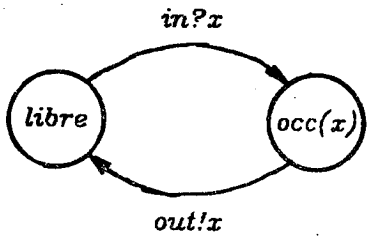
$$\mathcal{S} = (Q, A, T, s)$$

où:
-- Q est l'ensemble des états
-- A est l'ensemble des actions atomiques
-- $T \subseteq Q \times A \times Q$ est l'ensemble des transitions
-- $s \in Q$ est l'état initial.

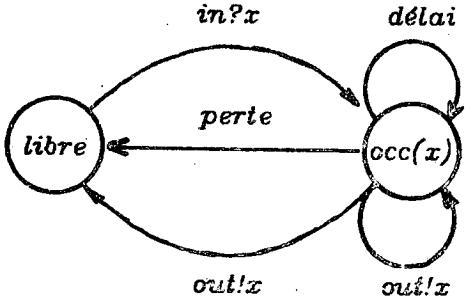
On a déjà utilisé la notation $q \xrightarrow{a} q'$; elle est équivalente (pour un système donné) à $(q, a, q') \in T$. On supposera implicitement que tous les états de Q sont accessibles à partir de

l'état initial s.

Exemple
On peut modéliser le comportement d'un médium de communication (tampon à une place) par un système de transitions comportant un état *libre*, à partir duquel on peut recevoir une donnée x, en exécutant une action $in?x$, ce qui amène dans un état "occupé par x", noté $occ(x)$; de cet état on peut délivrer la donnée en exécutant l'action $out!x$ et retourner dans l'état *libre*. On peut figurer ce système par:



De manière analogue on peut modéliser un médium moins fiable, qui peut introduire un délai d'acheminement arbitraire, des duplications et des pertes de message par:



Il est très souvent commode, lorsqu'on veut vérifier certaines propriétés d'un système, de faire abstraction de certains comportements. Précisons ce que cela signifie pour nous: en premier lieu, nous ferons l'hypothèse qu'on ne peut connaitre un état d'un système que par ses possibles reconfigurations (actions exécutées et ce que l'on peut connaitre de l'état atteint). C'est dire que nous suivons ici le point de vue de Milner [12], utilisant la notion de "bisimulation" attribuée à Park. En second lieu, l'abstraction porte sur les actions elles-mêmes: on peut par exemple considérer que certaines séquences d'actions représentent toutes la même action abstraite, ou que d'autres sont non pertinentes. Tout cela va être formalisé dans la notion de *congruence* de système (cf [4]), une généralisation immédiate de celle de bisimulation.

Quelques **notations:**
comme il est usuel, A^* désigne l'ensemble des mots sur A (suites finies d'actions), ε étant le mot vide, et A^+ l'ensemble des mots non vides. Nous noterons $u.v$ la concaténation de v après u. On peut alors étendre aux suites d'actions la relation de transition T dans un système donné:

(*i*) $q \overset{\varepsilon}{\Rightarrow} q$

(*ii*) $q \overset{a;u}{\Rightarrow} q' \Leftrightarrow \exists q''\ \ q \overset{a}{\Rightarrow} q''$ *et* $q'' \overset{u}{\Rightarrow} q'$ *(pour* $a \in A$*)*

On peut même l'étendre aux ensembles de suites d'actions (avec la même notation):

(*iii*) *pour* $U \subseteq A^*\ \ q \overset{U}{\Rightarrow} q' \Leftrightarrow \exists u \in U\ \ q \overset{u}{\Rightarrow} q'$

définition 1: critère d'abstraction

un critère d'abstraction sur un ensemble A d'actions est une *partition partielle* sur A^* c'est-à-dire un ensemble $C = \{e_i / i \in I\}$ d' "actions abstraites" $e_i \subseteq A^*$ deux à deux disjointes ▪

Exemples:
Les deux exemples les plus connus sont le "critère fort", pour lequel chaque action est distinguée (ie $C = \{\{a\} / a \in A\}$), et le critère d'observation de CCS, pour lequel les communications internes sont invisibles (cf [10,12]). Ce sont deux cas particuliers de critères d'abstraction obtenus à partir de **projections:** soit $V \subseteq A$ un ensemble d' *actions visibles*. Deux séquences sont *observationnellement équivalentes (modulo V)* si elles ont même contenu visible; les actions abstraites dans le critère $O(V)$ considéré sont les classes d'actions dans la relation d'équivalence:

$$u \equiv_V v \;\Leftrightarrow\; \mu_V(u) = \mu_V(v)$$

où $\mu_V : A^* \to V^*$ est la *projection* sur V^* (ie le morphisme qui efface les actions qui ne sont pas dans V).
Ces critères $O(V)$ obtenus en ignorant des actions sont eux-mêmes des cas particuliers de critères liés à certaines *congruences* du monoide libre A^*. Soit $W \subseteq A^* \times A^*$ une relation entre séquences d'actions et $\cong_W$ la congruence du monoide A^* engendrée par W; cette équivalence $\cong_W$ est la plus grossière qui satisfait:

(i) $W \subseteq \cong_W$

(ii) $u \cong_W u' \;\&\; v \cong_W v' \;\Rightarrow\; u;v \cong_W u';v'$

Elle détermine un critère, que nous noterons $\Omega(W)$, dont les actions abstraites sont les classes d'équivalences d'actions pour la relation $\cong_W$. Par exemple

$$\textit{pour } V \subseteq A : \; O(V) = \Omega(W) \quad \text{où } W = \{(a, \varepsilon) / a \notin V\}$$

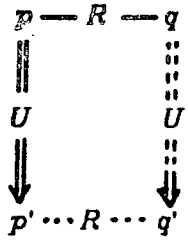
Ce type de critère est particulièrement important; c'est celui que nous utiliserons dans l'exemple de vérification traité plus loin (voir aussi l'appendice). On dira que W est ***alphabétique*** si $W \subseteq (A \cup \{\varepsilon\}) \times (A \cup \{\varepsilon\})$.

définition 2: congruence de système de transitions

une *congruence* sur un système $\theta = (Q, A, T, s)$ est un couple $\rho = (C, R)$ où:
(i) C est un critère d'abstraction sur A
(ii) R est une relation d'équivalence sur Q compatible avec les transitions modulo C, c'est-à-dire qui satisfait:

$$U \in C \;\textit{et}\; (p,q) \in R \;\textit{et}\; p \overset{U}{\Rightarrow} p' \;\Rightarrow\; \exists q' \; q \overset{U}{\Rightarrow} q' \;\textit{et}\; (p',q') \in R \;\blacksquare$$

Cette dernière propriété ("invariance" pour Brookes et Rounds [6], on pourrait aussi dire C-bisimulation) est habituellement dessinée:



Nous noterons par $[\![q]\!]_R$ la classe d'un état q dans la relation R.

définition 3: quotient d'un système

soit $\rho = (C, R)$ une congruence d'un système $\Theta = (Q, A, T, s)$. Le quotient de Θ par ρ est le système:

$$\Theta/\rho = (Q/R, C, \hat{T}, [\![s]\!]_R)$$

où

$$([\![p]\!]_R, U, [\![q]\!]_R) \in \hat{T} \Leftrightarrow \exists q' \; p \overset{U}{\Rightarrow} q' \; et \; q' \in [\![q]\!]_R \quad \blacksquare$$

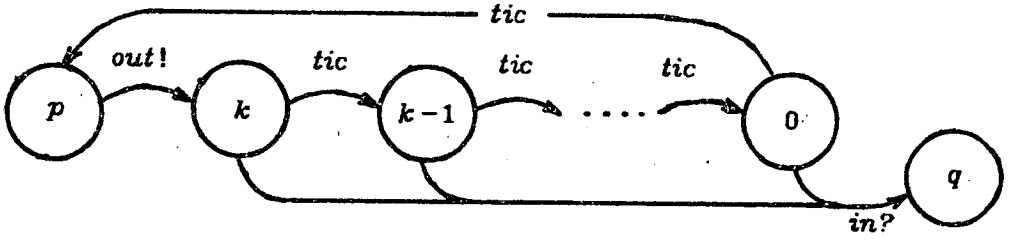
Il est facile de démontrer que pour chaque critère d'abstraction C sur un ensemble d'actions donné, il existe une relation d'équivalence *compatible* avec ce critère (ie qui est une C-bisimulation) qui est plus grossière que toutes les autres; nous l'avons appelée C-**équipollence** dans [4], et notée $\sim_C$ ($\sim$ lorsque C est le critère fort $O(A)$).

définition 4: équivalence de systèmes

deux systèmes $\Theta = (Q, A, T, s)$ et $\Theta' = (Q', A, T', s')$ sur le même ensemble d'actions A sont *C-équivalents* si leur quotient par $(C, \sim_C)$ sont isomorphes $\blacksquare$

Exemple

Considérons le système:



On peut le voir comme réalisant une "garde temporelle": étant dans l'état p, on passe dans un état d'où l'on "décompte un certain délai" (par *tic*), ou bien on accepte une "sortie normale" (par *in?* vers q). Si l'on "ignore le temps", on peut adopter le critère d'abstraction $O(V)$ (avec $V = \{out!, in?\}$) dont les actions abstraites sont:

$$\tau = \{tic\}^*$$

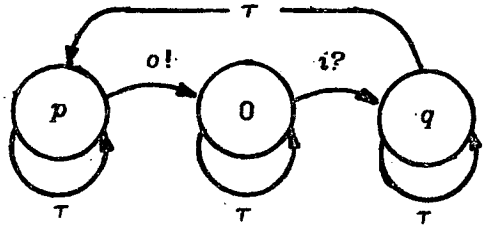
$$o! = \{tic\}^*; out!; \{tic\}^*$$

$$i? = \{tic\}^*; in?; \{tic\}^*$$

Alors on peut voir qu'avec la relation R dont les classes sont

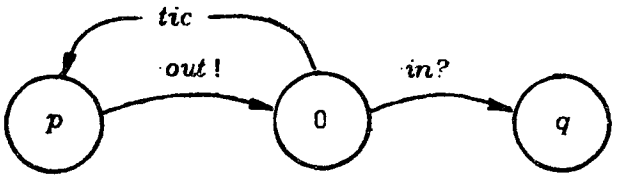
$$\{\{p\}, \{k, k-1, \ldots, 0\}, \{q\}\}$$

on obtient une congruence $\rho = (O(V), R)$, qui n'est autre, pour ce système, que $\sim_{O(V)}$. Le quotient par cette congruence peut être dessiné comme suit:



Bien entendu dans ce "modèle réduit", il est impossible de voir le "temps passé dans un état"

(actions τ). Le lecteur vérifiera que le système de garde temporelle précédent est $O(V)$-équivalent au système semblable mais à délai 0, déssiné ci-dessous:



3. Calculs de processus

Ce qui nous intéresse maintenant est la façon de *composer* des systèmes, et donc aussi de les *désigner* par des termes d'une algèbre. Pour cela nous nous donnons un ensemble (fini) F de symboles d'opérateurs; ces opérateurs vont représenter des "mécanismes de synchronisation". Nous décrivons la manière dont ils opèrent sur les systèmes de transitions par des *règles* qui spécifient les comportements possibles d'un système composite. Chaque règle (il y en a un nombre fini pour chaque opérateur $f \in F$) indique:

-- quels sont les composants d'un système (composé par f) qui sont autorisés à exécuter une action

-- quelles sont les actions autorisées, et

-- quelle transition le système effectue (action et état atteint).

On retrouve ces traits dans les règles que R. de Simone a étudiées dans [16,17], qui prennent la forme suivante:

$$\frac{x_{i_1} \xrightarrow{u_1} x'_{i_1}, \ldots, x_{i_k} \xrightarrow{u_k} x'_{i_k} \ , \ (u_1, \ldots, u_k, u) \in U}{f(x_1, \ldots, x_n) \xrightarrow{u} t}$$

qu'on peut lire:

si $\ldots x_{i_j} \xrightarrow{u_j} x'_{i_j} \cdots$ *et* $(u_1, \ldots, u_k, u) \in U$

alors $f(x_1, \ldots, x_n) \xrightarrow{u} t$

Dans une telle règle, $U \subseteq A^{k+1}$ est un prédicat sur les actions (on suppose donné l'ensemble A des actions), et t est un termes de l'algèbre considérée. De plus les conditions suivantes sont requises:

- $x_1, \ldots, x_n, x'_{i_1}, \ldots, x'_{i_k}$ sont des variables distinctes et $\{x_{i_1}, \ldots, x_{i_k}\} \subseteq \{x_1, \ldots, x_n\}$
- si l'on pose:

$$x'_j = \begin{cases} x'_{i_l} & \text{si } j = i_l \text{ pour un certain } l \ (1 \leq l \leq k) \\ x_j & \text{sinon} \end{cases}$$

alors t est un terme dans lequel n'apparaissent que les variables x'_j, chacune au plus une fois.

Ceci correspond au fait que pour déterminer l'évolution d'un système tel que $f(t_1, \ldots, t_n)$, on ne peut "aller voir dans le futur" de ses composants, ni les dupliquer.

Une **algèbre d'agents** (relative à un ensemble d'actions A) $\mathcal{A}_A$ est donnée par un ensemble (fini) de règles pour un ensemble (fini) d'opérateurs. Chaque terme (sans variable) que l'on peut écrire avec ces opérateurs représente un état dans un système de transitions: les transitions sont celles qui admettent une preuve selon les règles. Un **calcul de processus** est une algèbre d'agents munie d'une congruence $\rho = (C, R)$ sur le système de transitions entre termes; ceci constitue la *sémantique* du calcul, chaque classe de terme étant un *pro-*

cessus. Il est très agréable que cette congruence soit *aussi* une congruence d'algèbre, compatible avec les opérateurs, c'est-à-dire satisfaisant;

$$t_1 \equiv_R t'_1, \ldots, t_n \equiv_R t'_n \Rightarrow f(t_1, \ldots, t_n) \equiv_R f(t'_1, \ldots, t'_n)$$

En effet cela permet de parler des processus en faisant des raisonnements *équationnels* sur les termes qui les désignent, et de décomposer les preuves.

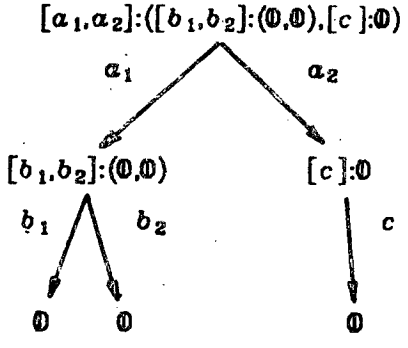
Exemple:
à l'ensemble A d'actions on peut associer des opérateurs de **choix gardé** (empruntés à [7], et issus de CSP): pour chaque n-uple $(a_1, \ldots, a_n)$ d'actions, on introduit un opérateur $[a_1, \ldots, a_n]$: à n arguments dont la sémantique opérationnelle est spécifiée par la règle (sans hypothèse sur les arguments)

$$\frac{\exists i \ (1 \leq i \leq n) \ a = a_i}{[a_1, \ldots, a_n]:(x_1, \ldots, x_n) \xrightarrow{a} x_i}$$

On désignera par $\mathcal{F}_A$ l'algèbre d'agents obtenue.

Remarques:
-- Lorsque $n=0$ on obtient une constante, que l'on notera $\mathbb{0}$ (NIL dans CCS), qui est un terme qui ne peut exécuter aucune transition.

-- On pourra aussi utiliser la notation plus classique $\sum_{1 \leq i \leq n} a_i:x_i$ ou encore $a_1:x_1 + \cdots + a_n:x_n$ pour ces opérateurs; cependant on évitera d'utiliser explicitement la somme (non gardée, c'est-à-dire $p+q$, comme dans CCS [10] ou SCCS [11,12,14]). La raison en est que cette dernière n'est pas toujours compatible avec les congruences de systèmes de transitions, alors que toute équipollence $\sim_{\Omega(W)}$ est compatible avec les choix gardés.
-- Lorsque $n=1$ on obtient un opérateur que nous appelerons **garde** et que nous noterons simplement $a:p$ ($a.p$ dans CCS). On a par exemple:



Le terme considéré ici aurait pu être noté: $a_1:(b_1:\mathbb{0}+b_2:\mathbb{0})+a_2:c:\mathbb{0}$

On peut étendre toute algèbre d'agents $\mathcal{A}_A$ en $\mathcal{A}_A^{rec}$, en lui adjoignant les opérateurs de **définition récursive:**

$$(t \text{ where } x_1 = t_1, \ldots, x_n = t_n)$$

Dénotons par $\theta[\theta_1/x_1, \ldots, \theta_n/x_n]$ le résultat de la substitution des termes $\theta_1, \ldots, \theta_n$ aux variables $x_1, \ldots, x_n$ dans le terme θ. Alors on peut spécifier la sémantique opérationnelle de ces opérateurs par les règles:

$$\frac{t \xrightarrow{a} t'}{(t \text{ where } x_1=t_1, \ldots, x_n=t_n) \xrightarrow{a} (t' \text{ where } x_1=t_1, \ldots, x_n=t_n)}$$

$$\frac{(t[t_1/x_1, \ldots, t_n/x_n] \text{ where } x_1=t_1,\ldots,x_n=t_n) \xrightarrow{a} t'}{(t \text{ where } x_1=t_1, \ldots, x_n=t_n) \xrightarrow{a} t'}$$

(une spécification différente, mais équivalente, est donnée dans [2,4]. Le lecteur aura rétabli le prédicat qui est omis dans ces règles).
Mentionnons ici le fait que ces opérateurs sont compatibles avec l'équipollence forte ~ (ie $\sim_{\Omega(A)}$); en particulier on a le loi (valable aussi pour toute autre équipollence):
L1: point fixe

en posant: $\theta_i = (x_i \text{ where } x_1=t_1, \ldots, x_n=t_n)$ *pour* $1 \leq i \leq n$

$t_i[\theta_1/x_1, \ldots, \theta_n/x_n] \sim \theta_i$

On a aussi, pour toute relation alphabétique W sur A^* (ie $W \subseteq (A \cup \{\varepsilon\}) \times (A \cup \{\varepsilon\})$, voir ci-dessus):

L2: solution unique

si *pour tout* j $(1 \leq j \leq n)$ $t_j \sim_{\Omega(W)} \sum_{1 \leq i \leq n} a_j^i : t_i$ *et* $a_j^i \notin_W \varepsilon$

alors $t_j \sim_{\Omega(W)} (x_j \text{ where } x_1 = \sum_{1 \leq i \leq n} a_1^i : z_i, \ldots, x_n = \sum_{1 \leq i \leq n} a_n^i : x_i)$

Milner a donné dans [13] un système de preuve complet pour l'équipollence forte des systèmes de transitions finis, décrits avec une syntaxe légèrement différente de celle de $\mathcal{F}_A^{rec}$ (voir aussi [3]).

Exemple:
l'algèbre $\mathcal{F}_A^{rec}$ obtenue à partir de la précédente permet de décrire tous les systèmes de **transitions finis** (sur l'ensemble d'actions A), c'est-à-dire les systèmes dont la relation de transition est finie. On peut ainsi décrire le premier système donné dans l'exemple 1, en négligeant la valeur des données, par:

$(x \text{ where } x = in? : y, y = out! : x)$

De même pour le second, si l'on admet qu'il y a deux valeurs 0 et 1 (bit de contrôle):

$med = (x \text{ where } x = in?0 : y_0 + in?1 : y_1$

$y_0 = delai : y_0 + perte : x + out!0 : y_0 + out!0 : x$

$y_1 = delai : y_1 + perte : x + out!1 : y_1 + out!1 : x)$

4. Parallélisme et synchronisation

Jusqu'à maintenant nous n'avons pas vraiment parlé de systèmes parallèles, même si l'on a évoqué une notion de "mécanismes de synchronisation". Pour rendre compte du parallélisme, il faut au moins pouvoir parler d'action globale d'un sytème, provenant de l'activité de ses composants agissant ensemble. Comme plusieurs sous-systèmes peuvent exécuter la même action ensemble, une action globale sera un *multi-ensemble* d'actions, où un élément peut avoir plusieurs occurrences. Une telle action globale est en soi atomique, sur le plan temporel, mais elle peut être composée d'actions élémentaires "simultanées". Si Act désigne l'ensemble des actions élémentaires, on dénotera par $\mathbb{M}(Act)$ l'ensemble des actions globales (multi-ensembles) composées à partir de Act. Nous adopterons ici une notation multiplicative, c'est-à-dire qu'un multi-ensemble tel que, par exemple

$\{ a , a , a , b , c , c \}$

sera noté

$a^3.b.c^2$

(l'élément neutre étant noté 1).

On retrouve ici l'idée de Milner [11,12] que les actions d'un système parallèle forment un *monoide commutatif*. Cette idée permet de rendre compte très naturellement du comportement de certains systèmes. Un exemple typique est celui du *tas*:

un tas, contenant k "jetons", peut simultanément
-- accepter, en exécutant des actions élémentaires a, un nombre quelconque de jetons,
-- délivrer, en exécutant b, au plus k jetons.

Par conséquent un tas peut être représenté par le système de transitions:

$$Tas = (\mathbb{N} , \mathbb{M}(\{a , b\}) , T)$$

(où $\mathbb{N}$ est l'ensemble des entiers), avec:

$$T = \{(k) \overset{a^n b^m}{\rightarrow} (k+n-m) \;/\; n \in \mathbb{N}, m \in \mathbb{N}, n+m \geq 0, m \leq k \}$$

Ce système est un cas particulier de ce que nous avons appelé dans [4] le système de transitions (asynchrone) déterminé par une **machine à places rationnelle** (cette classe de machines généralise directement les réseaux de Petri, voir [4,5]).

Bien entendu, dans l'optique "calcul de processus", le parallélisme va s'exprimer par une "loi de composition", exploitant l'idée que les actions forment un monoide commutatif M. Nous avons choisi dans le calcul MEIJE [2,4] une **composition parallèle** "asynchrone", où les composants sont indépendants. Ceci se traduit par la spécification suivante pour l'opérateur $\|$:

$$\frac{p \overset{a}{\rightarrow} p'}{(p \parallel q) \overset{a}{\rightarrow} (p' \parallel q)} \qquad \frac{q \overset{b}{\rightarrow} q'}{(p \parallel q) \overset{b}{\rightarrow} (p \parallel q')}$$

$$\frac{p \overset{a}{\rightarrow} p' \;,\; q \overset{b}{\rightarrow} q'}{(p \parallel q) \overset{a.b}{\rightarrow} (p' \parallel q')}$$

Il est facile de voir que cet opérateur est, pour la congruence forte, commutatif et associatif, et admet $\mathbb{0}$ pour élément neutre. Milner quant à lui a choisi de fonder son calcul SCCS [11,12] sur le *produit synchrone* qui est un opérateur déterministe satisfaisant la règle de comportement:

$$\frac{p \overset{a}{\rightarrow} p' \;,\; q \overset{b}{\rightarrow} q'}{(p \times q) \overset{a.b}{\rightarrow} (p' \times q')}$$

Un autre mode de composition bien connu est l' **entrelacement** (interleaving), qui interdit les comportements simultanés:

$$\frac{p \overset{a}{\rightarrow} p'}{(p \mid q) \overset{a}{\rightarrow} (p' \mid q)} \qquad \frac{q \overset{b}{\rightarrow} q'}{(p \mid q) \overset{b}{\rightarrow} (p \mid q')}$$

Concernant les primitives de synchronisation, nous avons introduit dans MEIJE le **pilotage** (ticking), noté $a * p$ qui a pour effet de multiplier les actions successives du système p par l'action donnée a:

$$\frac{p \xrightarrow{b} p'}{a * p \xrightarrow{a.b} a * p'}$$

On peut définir symétriquement une primitive de "désynchronisation" ∇, qui introduit à tout moment un délai arbitraire dans le comportement d'un agent:

$$\frac{}{\nabla(p) \xrightarrow{1} \nabla(p)} \qquad \frac{p \xrightarrow{a} p'}{\nabla(p) \xrightarrow{a} \nabla(p')}$$

Il reste un aspect concernant les actions dont il faut parler, c'est celui de la communication. Là encore nous allons suivre Milner, pour qui l'acte primitif de communication est la "poignée de main" (ou rendez-vous, à deux). Ici la formalisation de la notion d'action globale non-interruptible, par la structure de monoide, se révèle très commode. En effet la communication synchronisée se traduit simplement par la présence d'actions élementaires inverses les unes des autres. Les actions globales seront donc comme précédemment des produits d'actions élémentaires, certaines (qui sont des échanges de signaux) avec un exposant positif ("émission") ou négatif ("réception"). Si l'on désigne par Sig l'ensemble des noms de signaux (supposé disjoint de l'ensemble Act des "actions de calcul"), on dénotera par

$$\Gamma(Act, Sig)$$

le monoide commutatif d'actions obtenu. Par exemple si a, b sont dans Act, α, β dans Sig, on aura

$$a^2.a^{-2}.\beta.b^3 \in \Gamma(Act, Sig)$$

et

$$(a^2.a^{-2}.\beta.b^3).(c.a.\beta^{-2}.a^2) = a^3.b^3.c.\beta^{-1}$$

La loi (en notant s^- pour s^{-1})

$$s.s^- = 1 \quad pour\ s \in Sig$$

est la *loi d'interaction* (ou de *communication*) pour le monoide $\Gamma(Act, Sig)$.

Un opérateur de synchronisation très important peut alors être défini: c'est la **restriction** qui force des communications internes, et symétriquement interdit l'échange de certains signaux avec l'extérieur. La spécification de ces opérateurs, paramétrés par les noms de signaux, est:

$$\frac{p \xrightarrow{b} p' \ , \ b \in \Gamma(Act, Sig - \{\alpha\})}{p \setminus \alpha \xrightarrow{b} p' \setminus \alpha}$$

Ici le prédicat $b \in \Gamma(Act, Sig - \{\alpha\})$ signifie que b ne contient pas, de manière irréductible, d'émission ou de réception du signal α.

Les trois calculs constitués des opérateurs de *gardes*, *restrictions*, *définitions récursives*, et, respectivement:

-- la *composition parallèle* "asynchrone" et le *pilotage*
-- le *produit synchrone* et la *désynchronisation*
-- le *produit synchrone* et l' *entrelacement*

sont équivalents (modulo la congruence forte). Cela signifie que dans chacun d'eux les primitives des autres sont *définissables* (modulo la congruence forte) par une expression (voir [4]). Par exemple on a:

$$(p \times q) \sim (a^2 * (a^- * p \parallel a^- * q)) \setminus a$$

$$(p \mid q) \sim (a * (a^- * p \parallel a^- * q)) \setminus a$$

D'autres exemples d'opérateurs définissables ou dérivés dans ces calculs peuvent être trouvés dans [2,4]. R. de Simone a par ailleurs démontré (cf [16,17]) que si l'on rajoute à l'un de ces calculs les opérateurs de **renommage** (voir ci dessous), alors on obtient des calculs **universels** en ce sens que, modulo la congruence forte, on peut y exprimer n'importe quel système de transitions effectif (ie récursivement énumérable) et plus généralement n'importe quelle algèbre d'agents effective (les prédicats sur les actions apparaissant dans les règles étant récursivement énumérables).

Les opérateurs de renommage sont paramétrés par certains morphismes du monoide d'actions: si l'on se donne une liste $c_1, \ldots, c_n$ d'éléments de $Act \cup Sig$ qui seront respectivement renommés en les actions $u_1,\ldots,u_n$ de $\Gamma(Act, Sig)$ alors on peut définir un opérateur

$$<u_1/c_1,\ldots,u_n/c_n>p$$

qui va effectuer ce renommage sur les actions successives de p. Soit

$$\lambda = <u_1/c_1,\ldots,u_n/c_n>$$

et φ_λ le morphisme déterminé par ce renommage . Par exemple si

$$\lambda = < \gamma b \,/\, b \,,\, \gamma\beta \,/\, \alpha >$$

on a

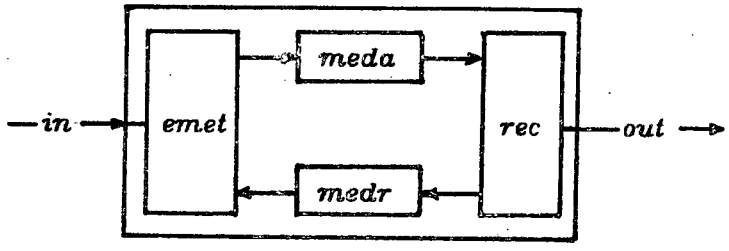
$$\begin{aligned}\varphi_\lambda(\, b^2\alpha^-\beta\,) &= \varphi_\lambda(b)\varphi_\lambda(b)\varphi_\lambda(\alpha)^-\varphi_\lambda(\beta)\\ &= \gamma b\,\gamma b\,\gamma^-\beta^-\beta\\ &= \gamma b^2\end{aligned}$$

Alors la spécification de $\lambda\, p$ est:

$$\frac{p \overset{a}{\rightarrow} p'}{\lambda p \overset{\varphi_\lambda(a)}{\rightarrow} \lambda p'}$$

5. Un exemple de vérification

Dans ce paragraphe nous montrons sur un exemple comment on peut appliquer la théorie des calculs de processus au problème de la vérification. Nous avons choisi l'exemple standard dans ce domaine, c'est-à-dire une preuve de protocole, une version du protocole du bit alterné pour être précis (nous poursuivons ici l'étude entreprise dans [1]). La structure statique d'un tel système est généralement décrite par un réseau comportant des agents émetteur et récepteur qui communiquent à travers des médiums. On peut figurer ce réseau par:



On va faire ici abstraction des valeurs des messages transmis, ce qui est justifié dans la mesure où il n'y a pas de mémorisation; on ne retiendra donc que les valeurs 0 ou 1 du bit de contrôle b, qui transite de l'émetteur vers le médium aller ($meda$) à travers une porte em, de ce dernier vers le récepteur à travers mr, etc. Les signaux véhiculés à l'intérieur du système seront donc em_0, em_1, rm_0, rm_1...

Milner a montré dans [9] (voir [8] pour la relation entre MEIJE et les expressions de réseaux) qu'un tel dessin peut se décrire par une expression construite, à partir des agents, avec les opérateur de renommage, composition parallèle et restriction, ici:

$$Sys = (em \parallel meda \parallel medr \parallel rec) \setminus S$$

où S est l'ensemble des signaux internes:

$$S = \{em_b, mr_b, rm_b, me_b / b \in \{0,1\}\}$$

Chaque agent quant à lui est décrit par un sytème de transition fini, donc par un terme de l'algèbre $\mathcal{F}_M^{rec}$, où M est le monoide:

$$M = \Gamma(A, S)$$

(nous décrirons A plus loin)

Dans le problème qui nous intéresse, les algorithmes émetteur et récepteur sont chargés de transmettre correctement des messages à travers des lignes qui peuvent introduire des délais d'acheminement, des duplications, des pertes non signalées. Par conséquent chaque médium est un exemplaire d'un système déjà rencontré ci-dessus (avec des notations légèrement différentes):

$$meda = \lambda_0\, med \quad \text{où} \quad \lambda_0 = \langle em_0/in_0, em_1/in_1, mr_0/out_0, mr_1/out_1 \rangle$$
$$medr = \lambda_1\, med \quad \text{où} \quad \lambda_1 = \langle rm_0/in_0, rm_1/in_1, me_0/out_0, me_1/out_1 \rangle$$

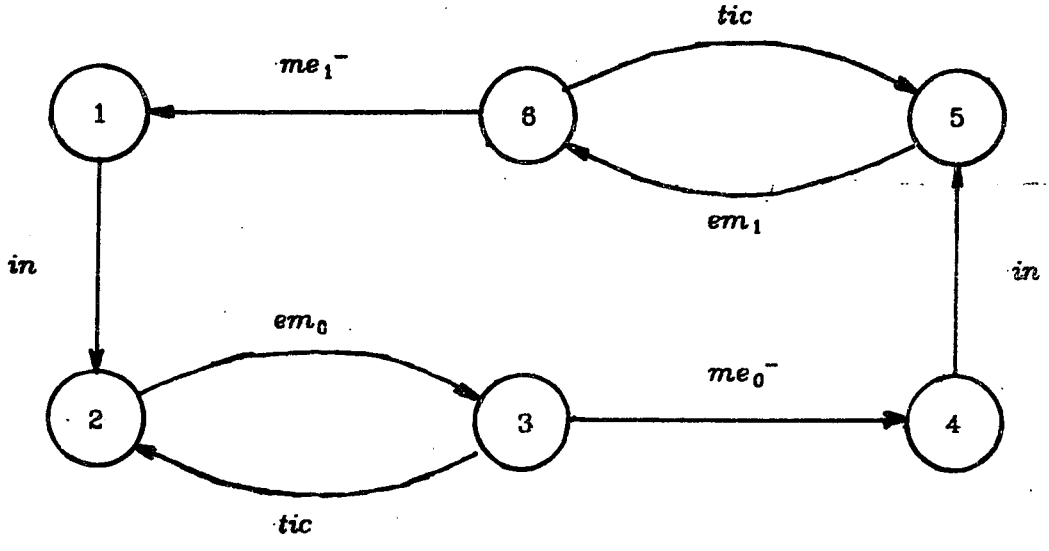
avec:

$$med = (m_1 \textbf{ where } m_1 = in_0^- : m_2 + in_1^- : m_3$$
$$m_2 = delai : m_2 + perte : m_1 + out_0 : m_2 + out_0 : m_1$$
$$m_3 = delai : m_3 + perte : m_1 + out_1 : m_3 + out_1 : m_1)$$

On notera aussi:

$$med = (m_1 \textbf{ where } MED)$$

La logique des algorithmes émetteur et récepteur est bien connue: dans le sens "retour", le bit de contrôle joue le rôle d'accusé de réception, et l'émetteur n'accepte de nouveau message (par in) qu'après avoir reçu un accusé correct. De plus nous supposerons que dans son état d'attente d'un accusé, l'émetteur déclenche une garde temporelle à l'expiration de laquelle il réémet le message. Dans la vérification du protocole nous ferons abstraction du temps passé à l'intérieur du système entre l'entrée et la sortie d'un message; on peut par conséquent supposer (voir ci-dessus) que la garde temporelle est à délai 0. L'émetteur est alors le système:



La table de transitions de cet algorithme est représentée par le système d'équations:

$$EM = \begin{cases} e_1 = in:e_2 \\ e_2 = em_0:e_3 \\ e_3 = me_0^-:e_4 + tic:e_2 \\ e_4 = in:e_5 \\ e_5 = em_1:e_6 \\ e_6 = me_1^-:e_1 + tic:e_5 \end{cases}$$

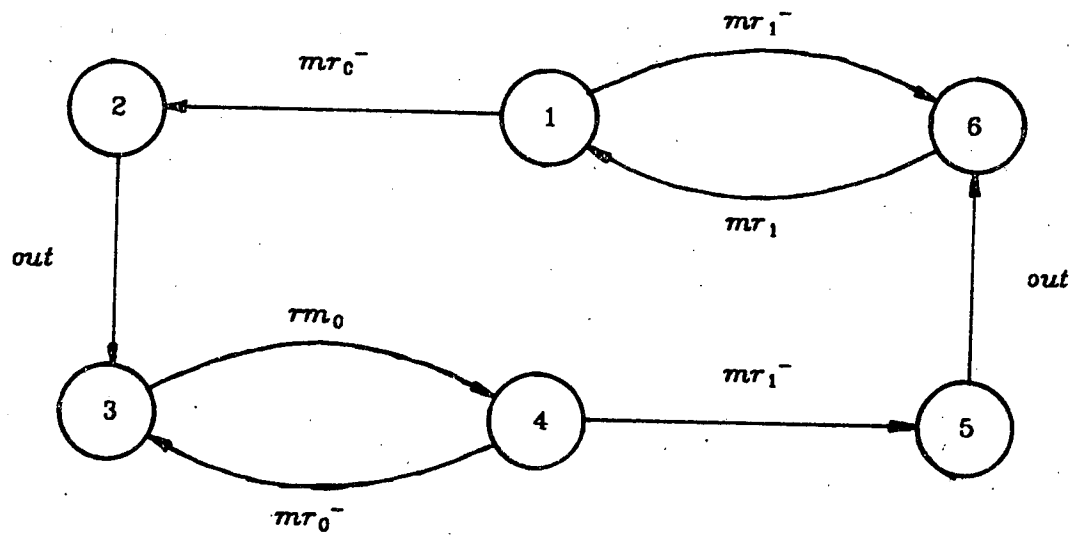
Un terme qui permet de décrire l'émetteur est alors le suivant:

$$emet = (e_1 \textbf{ where } EM)$$

On peut maintenant préciser quel est l'ensemble A d'actions élémentaires intervenant dans le système:

$$A = \{in, out, tic, delai, perte\}$$

La logique du récepteur est tout à fait similaire à celle de l'émetteur, comme on peut en juger sur son schéma:



Le lecteur n'aura donc pas de mal à écrire un système d'équations REC en les variables $r_1,\ldots,r_6$, de telle sorte que:

$$rec = (r_1 \textbf{ where } REC)$$

Pour simplifier les notations, nous poserons:

$$em_i = (e_i \textbf{ where } EM) \quad (1 \leq i \leq 6)$$

$$meda_j = \lambda_0\, med_j \quad (1 \leq j \leq 3)$$

$$medr_k = \lambda_1\, med_k \quad (1 \leq k \leq 3)$$

$$med_l = (m_l \textbf{ where } MED) \quad (1 \leq l \leq 3)$$

$$rec_h = (r_h \textbf{ where } REC) \quad (1 \leq h \leq 6)$$

et

$$[i,j,k,h] = (em_i \,||\, meda_j \,||\, medr_k \,||\, rec_h) \backslash S$$

(il est possible de voir là la disposition de "jetons" dans les états de chacun des composants de Sys). L'état initial du système est donc $[1,1,1,1]$.

Le sytème à analyser étant maintenant décrit, nous devons préciser quelles seront les *conditions et objectifs de la vérification*. En fait nous les avons plus ou moins déjà indiqués: il s'agit de vérifier que, si l'on néglige les actions internes du système, celui-ci rend bien le service qu'on en attend. Par conséquent nous adopterons pour regarder Sys le critère d'observation $\Omega(W)$ pour lequel les actions pertinentes sont celles qui contiennent *in* ou *out*. Toute action w de $\Gamma(A,S)$ peut se décomposer en $w = u.v$ avec $u \in \mathbb{M}(\{in,out\})$ et $v \in \Gamma(\{tic,delai,perte\},S)$; on décide alors que son contenu observable est u, et que 1 n'est pas visible. C'est dire que W est décrit de la façon suivante:

$$W = \{(v,\varepsilon) / v \in \Gamma(\{tic,delai,perte\},S)\}$$
$$\cup \{(u.v,u) / u \in \mathbb{M}(\{in,out\}), v \in \Gamma(\{tic,delai,perte\},S)\}$$

(c'est une relation alphabétique)
Le but de la vérification est alors de démontrer la

Proposition:

pour le critère d'abstraction $\Omega(W)$ le protocole Sys est équivalent au service décrit par $Service = (libre \textbf{ where } libre = in:occ\ ,\ occ = out:in)$ •

La preuve utilise deux types d'arguments: raisonnement équationnel sur les termes d'une part, raisonnement "comportemental" de l'autre. Mais nous n'avons pas encore précisé la syntaxe des termes que nous utiliserons: ce sera celle de l'algèbre $\mathcal{C}_M(\mathcal{F}_M{}^{rec})$ formée en utilisant les opérateurs de *choix gardés*, *composition parallèle*, *restrictions* et *renommages* appliqués aux agents de $\mathcal{F}_M{}^{rec}$ qui décrivent les systèmes de transitions finis (voir ci-dessus). Nous aurions pu adopter une autre syntaxe, celle de [7] par exemple; bien entendu, les agents obtenus dénotent toujours des systèmes finis.

Les lois que nous utiliserons dans le raisonnement équationnel seront, outre $L1$ et $L2$, les suivantes:

L3: associativité

$$((p \parallel q) \parallel r) \sim (p \parallel (q \parallel r))$$

L4: distribution

$$\Big(\sum_{1\le i\le n} a_i:p_i \parallel \sum_{1\le j\le k} b_j:q_j\Big) \sim \sum_{1\le i\le n} a_i:(p_i \parallel \Big(\sum_{1\le j\le k} b_j:q_j\Big)) + \sum_{1\le i\le n}\sum_{1\le j\le k} (a_i b_j):(p_i \parallel q_j) + \sum_{1\le j\le k} b_j:(\Big(\sum_{1\le i\le n} a_i:p_i\Big) \parallel q_j)$$

L5: restriction

$$\Big(\sum_{1\le i\le n} b_i:p_i\Big) \setminus \alpha \sim \sum_{1\le j\le k} c_j : (p_j \setminus \alpha)$$

$$\text{où} \quad \{c_1,\ldots,c_k\} = \{b_1,\ldots,b_n \,/\, b_i \in \Gamma(A, S-\{\alpha\})\}$$

Ces trois premières lois sont l'analogue du *théorème d'expansion* de Milner ([10]) qui permet de trouver les équations d'un système parallèle à partir de celles de ses composants.

L6: renommage

$$\lambda\Big(\sum_{1\le i\le n} a_i:p_i\Big) \sim \sum_{1\le i\le n} \varphi_\lambda(a_i):\lambda\, p_i$$

L7: commutativité des choix

$$\sum_{1\le i\le n} a_i:p_i \sim \sum_{1\le i\le n} a_{\sigma(i)}:p_{\sigma(i)}$$

où σ est une permutation de $\{1,\ldots,n\}$

Ces lois sont valables en fait pour n'importe quelle équipollence, puisque ~ est la plus fine. D'autres lois sont spécifiques des équipollences $\sim_{\Omega(\mathcal{W})}$, comme:

L8: idempotence

$$\textbf{si } a \cong_{\mathcal{W}} b \quad \textbf{alors} \quad a:p + b:p + \sum_{1\le i\le n} a_i:p_i \sim_{\Omega(\mathcal{W})} a:p + \sum_{1\le i\le n} a_i:p_i$$

L9: absorption

$$\textbf{si } a \cong_{\mathcal{W}} \varepsilon \quad \textbf{alors} \quad a:p \sim_{\Omega(\mathcal{W})} p$$

Ces dernières propriétés peuvent être vues comme des *simplifications*. Elle mettent en jeu les comportements des états (termes), introduits par les choix gardés. Dans le même esprit, il y a une propriété qui nous sera fort utile, exprimée par:

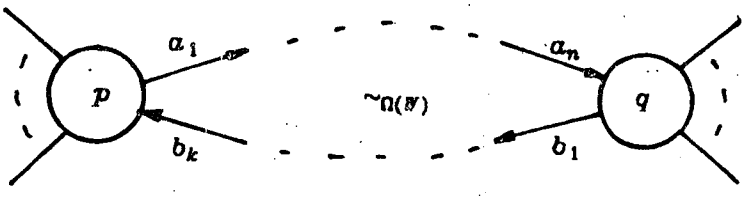
lemme *L:*

si deux états p et q d'un système de transitions sont sur un même cycle de transitions invisibles, c'est-à-dire:

$$\exists a_1,\ldots,a_n \cong_{\mathcal{W}} \varepsilon \;\; \exists b_1,\ldots,b_k \cong_{\mathcal{W}} \varepsilon \quad p \xrightarrow{a_1} \cdots \xrightarrow{a_n} q \quad \text{et} \quad q \xrightarrow{b_1} \cdots \xrightarrow{b_k} p$$

alors ils sont équivalents: $p \sim_{\Omega(\mathcal{W})} q$ ∎

On peut figurer cette situation de la façon suivante:



Cette propriété est particulièrement intéressante pour l'analyse des protocoles. En effet, un protocole est chargé de rétablir en cours de route des situations mal engagées du fait de mauvais fonctionnements des lignes sous-jacentes; par conséquent s'il s'écarte, à l'insu de l' utilisateur, du déroulement normal de la procédure, il *doit* pouvoir y revenir.

Le lemme L ne suffit pas tout à fait à assurer cela; c'est pourquoi nous utiliserons aussi

lemme *L':* **élimination des cycles**

si $p \sim \sum_{1\le i\le n} a_i:p_i + \sum_{1\le j\le k} b_j:q_j$ avec

$\forall i\ (1\le i\le n)\ \ a_i \cong_{\Omega(\mathcal{W})} \varepsilon$

$\forall i\ (1\le i\le n)\ \ p_i \overset{\varepsilon}{\Rightarrow} p$

$$\forall i\ \forall c\ \ p_i \overset{c}{\dashrightarrow} p' \ \Rightarrow \begin{cases} c \cong_{\Omega(\mathcal{W})} \varepsilon \ \ \& \ \ p' \in \{p, p_1,\ldots,p_n\} & \textit{ou bien} \\ \exists j\ c \cong_{\Omega(\mathcal{W})} b_j \ \ \& \ \ p' \sim_{\Omega(\mathcal{W})} q_j \end{cases}$$

alors $p \sim_{\Omega(\mathcal{W})} \sum_{1\le j\le k} b_j:q_j$ ∎

C'est cette idée que l'on va suivre pour conduire la preuve de la proposition: on va suivre pas à pas le déroulement normal du protocole, celui où les médiums fonctionnent comme des lignes parfaites (ie comme *Service*). Ce déroulement normal est le suivant:

$$[1,1,1,1] \xrightarrow{in} [2,1,1,1] \xrightarrow{1} [3,2,1,1] \xrightarrow{1} [3,1,1,2] \xrightarrow{out} [3,1,1,3] \xrightarrow{1} [3,1,2,4] \xrightarrow{1} [4,1,1,4]$$
$$\xrightarrow{in} [5,1,1,4] \xrightarrow{1} [6,3,1,4] \xrightarrow{1} [6,1,1,5] \xrightarrow{out} [6,1,1,6] \xrightarrow{1} [6,1,3,1] \xrightarrow{1} [1,1,1,1]$$

On va développer, grâce aux lois $L1, L3-L5$, les équations (modulo $\sim$) des douze états rencontrés. Puis, à chaque pas, on simplifiera les équations soit directement par $L9$, soit en appliquant le lemme d'élimination des cycles; dans ce cas il restera encore à simplifier par $L7-L9$ et grâce au lemme L pour se rapprocher (modulo $\sim_{\Omega(W)}$) des équations du *Service*. On concluera en utilisant $L2$. Ainsi, au lieu de construire puis réduire le système global (qui comporte de l'ordre de 150 états, et beaucoup plus de transitions, voir [1]), ou examine "ce qui se passe" autour d'une douzaine d'états. Dans toute cette preuve on use librement du fait que les équipollences $\sim$ et $\sim_{\Omega(W)}$ sont compatibles avec les opérateurs des algèbres $\mathcal{C}_M(\mathcal{F}_M{}^{rec})$ et $\mathcal{F}_M{}^{rec}$ respectivement.

On a

$$\text{par } L1: \quad em_1 \sim in:em_2$$
$$rec_1 \sim rm_0{}^-:rec_2$$
$$med_1 \sim in_0{}^-:med_2 + in_1{}^-:med_3$$

donc

$$\text{par } L6: \quad meda_1 \sim em_0{}^-:meda_2 + em_1{}^-:meda_3$$
$$medr_1 \sim rm_0{}^-:medr_2 + rm_1{}^-:medr_3$$

En utilisant le fait que $\sim$ est compatible avec les opérateurs et (plusieurs fois) $L3$ et $L4$, puis $L5$, on obtient une première équation:

$$[1,1,1,1] \sim in:[2,1,1,1]$$

Appliquant la même technique pour $[2,1,1,1]$, il vient:

$$[2,1,1,1] \sim (em_0.em_0{}^-):[3,2,1,1]$$
$$\sim 1:[3,2,1,1]$$

En vertu de $L9$ (et du fait que $\sim \subseteq \sim_{\Omega(W)}$):

$$[2,1,1,1] \sim_{\Omega(W)} 1:[3,2,1,1] \sim_{\Omega(W)} [3,2,1,1]$$

Développant l'équation de $[3,2,1,1]$ (toujours par $L1, L3-L6$) on trouve:

$$[3,2,1,1] \sim tic:[2,2,1,1] + perte:[3,1,1,1] + delai:[3,2,1,1]$$
$$+ (tic.perte):[2,1,1,1] + (tic.delai):[2,2,1,1]$$
$$+ 1:[3,2,1,2] + 1:[3,1,1,2] + tic:[2,2,1,2] + tic:[2,1,1,2]$$

La "suite normale" est $[3,1,1,2]$. C'est ici qu'il faut utiliser le lemme L' d'elimination des délais, avec pour p_i les états $[\ldots,\ldots,\ldots,1]$ où le récepteur n,a pas bougé, et pour q_j les états $[\ldots,\ldots,\ldots,2]$ où il a reçu le message. On vérifie en effet que, par exemple:

$$[2,2,1,1] \xrightarrow{perte} [2,1,1,1] \xrightarrow{1} [3,2,1,1] \quad et$$
$$[2,2,1,1] \sim delai:[2,2,1,1] + perte:[2,1,1,1] + 1:[2,2,1,2] + 1:[2,1,1,2]$$
$$(et \quad delai \cong_{\Omega(W)} perte \cong_{\Omega(W)} \varepsilon)$$

On obtient alors par le lemme L':

$$[3,2,1,1] \sim_{\Omega(W)} 1:[3,2,1,2] + 1:[3,1,1,2] + tic:[2,2,1,2] + tic:[2,1,1,2]$$

Mais grâce au lemme L, on voit facilement que:

$$[3,2,1,2] \sim_{\Omega(W)} [3,1,1,2] \sim_{\Omega(W)} [2,2,1,2] \sim_{\Omega(W)} [2,1,1,2]$$

Par conséquent, appliquant $L8$ (plusieurs fois) puis $L9$ il vient:

$$[3,2,1,1] \sim_{\Omega(W)} [3,1,1,2]$$

Poursuivant sur le même chemin, on trouve, avec le même type d'argument:

$$[3,1.1.2] \sim_{\Omega(W)} out:[3,1,1,3]$$

puis

$$[3,1,1,3] \sim_{\Omega(W)} [3,1,2,4] \sim_{\Omega(W)} [4,1,1,4]$$

En analysant les six premiers états sur le cheminement normal, on a analysé la moitié du protocole. Pour la partie symétrique correspondant à la valeur 1 du bit de contrôle on procède de la même manière, d'où:

$[1,1,1,1] \sim_{\Omega(W)} in:[3,1,1,2]$

$[3,1,1,2] \sim_{\Omega(W)} out:[4,1,1,4]$

$[4,1,1,4] \sim_{\Omega(W)} in:[6,1,1,5]$

$[6,1,1,5] \sim_{\Omega(W)} out:[1,1,1,1]$

La loi $L2$ nous indique alors que le sytème Sys est $\sim_{\Omega(W)}$-équivalent au terme t_0, avec

$$t_i = (x_i \textbf{ where } x_0 = in:x_1, x_1 = out:x_2$$
$$x_2 = in:x_3, x_3 = out:x_0)$$

La proposition est établie dès lors qu'on a remarqué que l'équivalence dont les classes sont

$$\{t_0, t_2\}, \{t_1, t_3\}$$

est une congruence (pour le critère fort) du système de transitions déterminé par le terme précédent, car alors ce terme est lui même équivalent au *Service*.

Remerciements

Les arguments utilisés dans une première version de la preuve ci-dessus étaient erronés, comme me l'a signalé G. Gonthier et je l'en remercie.

Références

[1] D. Austry: "Aspects syntaxiques de MEIJE, un calcul pour le parallélisme", Thèse de 3ème cycle, Université Paris 7 (1983).

[2] D. Austry & G. Boudol: "Algèbre de processus et synchronisation", Theoret. Comput. Sci. 30 (1984) 91-131.

[3] J.A. Bergstra & J.W. Klop: "A complete inference system for regular processes with silent moves", Report CS-R8420 (1984), CWI, Amsterdam.

[4] G. Boudol: "Notes on algebraic calculi of processes", Proc. of the Advanced Course on Logics and Models for Verification and Specification of Concurrent Systems (K. Apt Ed.), La colle-sur-Loup (1984). Rapport INRIA 395 (1985).

[5] G. Boudol, G. Roucairol & R. de Simone: "Petri nets and algebraic calculi of processes", soumis à "Advances in Petri Nets" (G. Rozenberg Ed) (version abrégée: STACS 85, LNCS 182, 59-70).

[6] S. Brookes & W.C. Rounds: "Behavioural equivalence relations induced by programming logics", ICALP 83, Lecture Notes in Comput. Sci. 154 (1983) 83-96.

[7] Ph. Darondeau & L. Kott: "On the observational semantics of fair parallelism", ICALP 83, Lecture Notes in Comput. Sci. 154 (1983) 147-159 (voir aussi Rapport INRIA 262).

[8] G. Gonthier: "Algebraic calculi of processes and net expressions", Rapport INRIA 367 (1985), à paraitre dans Theoret. Comput. Sci.

[9] R. Milner: "Flowgraphs and flow algebras", JACM 26 (1979)794-818.

[10] R. Milner: "A Calculus of Communicating Systems", Lecture Notes in Comput. Sci. 92 (1980).

[11] R. Milner: "On relating synchrony and asynchrony", Report CSR-75-80, Edinburgh Univ. (1980).

[12] R. Milner: "Calculi for synchrony and asynchrony", Theoret. Comput. Sci. 25 (1983) 267-310.

[13] R. Milner: "A complete inference system for a class of regular behaviours", J. of Computer and Systems Sci. 28(1984) 439-466.

[14] G. Plotkin: "A structural approach to operational semantics", Report Daimi FN-19, Comput. Sci. Dept., Aarhus Univ. (1981).

[15] R. de Simone: "Note on MEIJE and SCCS: infinite sum operators *vs* non-guarded definitions", Theoret Comput Sci. 30 (1984) 133-138.

[16] R. de Simone: "Calculabilité et expressivité dans l'algèbre de processus MEIJE", Thèse de 3ème cycle, Université Paris 7 (1984).

[17] R. de Simone: "Higher-level synchronizing devices in MEIJE-SCCS", Rapport INRIA 360 (1985), à paraitre dans Theoret Comput. Sci.

Appendice

Les équipollences $\sim_{\Omega(W)}$ admettent un *principe de preuve* (voir aussi [2,4]), grâce auquel (pour W alphabétique) on montre par exemple les lois $L2$, $L8$–$L10$. Pour

$$W \subseteq A^* \times A^*$$

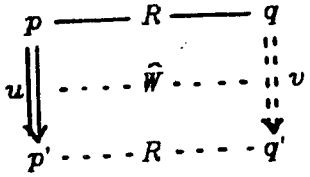
posons

$$\widehat{W} = W \cup \{(b,a)/(a,b) \in W\} \cup \{(a,a)/a \in A\}$$

(on a $\cong_{\widehat{W}} = \cong_W$).
Alors, avec ces notations:

lemme: principe de preuve

soit $\theta = (Q, A, T, s)$ un système de transitions sur A, $R \subseteq Q \times Q$ une équivalence entre les états.
Si (voir l'énoncé formel ci-dessous)



alors R est une $\Omega(W)$-bisimulation
donc $R \subseteq \sim_{\Omega(W)}$ ∎

Le diagramme ci-dessus représente l'énoncé formel:

$\forall p,q,p' \in Q, \forall u \in A^*$ *tel que* $\exists w\ (u,w) \in \widehat{W}$

si $(p,q) \in R$ & $p \overset{u}{\Rightarrow} p'$

alors $\exists q'\ \exists v$ *tels que* $(u,v) \in \widehat{W}$, $q \overset{v}{\Rightarrow} q'$ & $(p',q') \in R$

Cette proposition est une généralisation immédiate de la proposition 8.4 de Milner dans [12]; la preuve ne présente pas de difficulté.

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique